# Supporting Software Developers' Focused Work on Window-Based Desktops

Pilzer, Jan ; Rosenast, Raphael ; Meyer, André ; Huang, Elaine May ; Fritz, Thomas

Abstract: Software developers, like other information workers, continuously switch tasks and applications to complete their work on their computer. Given the high fragmentation and complexity of their work, staying focused on the relevant pieces of information can become quite challenging in today's windowbased environments, especially with the ever increasing monitor screen-size. To support developers in staying focused, we conducted a formative study with 18 professionals in which we examined their computer based and eye-gaze interaction with the window environment and devised a relevance model of open windows. Based on the results, we developed a prototype to dim irrelevant windows and reduce distractions, and evaluated it in a user study. Our results indicate that our model was able to predict relevant open windows with high accuracy and participants felt that integrating visual prominence into the desktop environment reduces clutter and distraction, which results in reduced window switching and an increase in focus.

DOI: https://doi.org/10.1145/3313831.3376285

Other titles: © 2020 Copyright is held by the owner/author(s). ACM ISBN 978-1-4503-6708-0/20/04. https://dx.doi.org/10.1145/3313831.3376285

# Supporting Software Developers' Focused Work on Window-Based Desktops

**Jan Pilzer**[1], **Raphael Rosenast**[2], **André N. Meyer**[3], **Elaine M. Huang**[3], **Thomas Fritz**[3]
[1]University of British Columbia, Vancouver, Canada
[2,3]University of Zurich, Zurich, Switzerland
[1]pilzer@cs.ubc.ca, [2]raphael.rosenast@uzh.ch, [3]{ameyer, huang, fritz}@ifi.uzh.ch

## ABSTRACT
Software developers, like other information workers, continuously switch tasks and applications to complete their work on their computer. Given the high fragmentation and complexity of their work, staying focused on the relevant pieces of information can become quite challenging in today's window-based environments, especially with the ever increasing monitor screen-size. To support developers in staying focused, we conducted a formative study with 18 professionals in which we examined their computer based and eye-gaze interaction with the window environment and devised a relevance model of open windows. Based on the results, we developed a prototype to dim irrelevant windows and reduce distractions, and evaluated it in a user study. Our results indicate that our model was able to predict relevant open windows with high accuracy and participants felt that integrating *visual prominence* into the desktop environment reduces clutter and distraction, which results in reduced window switching and an increase in focus.

### Author Keywords
Window Management; User Interfaces; Window Relevance; Focus; Productivity

### CCS Concepts
•**Human-centered computing** → **Interactive systems and tools;** *Graphical user interfaces;* Field studies;

## INTRODUCTION
Multi-tasking and fragmentation of work are known challenges of modern knowledge work [9, 15, 31]. While multi-tasking is necessary and beneficial in enabling information workers to make progress on more than one task, it can come at a significant cost: reduced quality and more errors [44], more time spent at a task overall [43, 44], increased stress [2, 27], and lower productivity [28, 32]. Although these difficulties are to some extent inherent and inevitable side effects of engaging in multiple work activities simultaneously, we believe that these effects can also be decreased if the information worker can focus only on the most *relevant* tasks at any given time, and is subjected to *limited distraction* from less relevant tasks.

Modern window-based computer systems provide support for multi-tasking by allowing information workers access and visibility to many applications and digital artifacts simultaneously. However, with increasing screen size, multi-monitor support and the ability to use and see several applications at the same time, these systems become more cluttered, thereby providing ample opportunity for distractions and switches to less relevant tasks and decreasing focus [1]. Additionally, the more virtually cluttered the computer is by having more applications, windows or tabs open, the higher the cognitive costs are [30] and the more time workers need to arrange them and to find what they are looking for [20, 38]. Switching between windows and tasks further diminishes information workers' concentration by leaving an "attention residue" that makes it more difficult to resume a task once a worker is distracted [23].

We address these challenges by investigating whether *visual prominence* designed into the desktop environment can be leveraged to reduce clutter and distraction. By supporting *strategic* multi-tasking and providing lightweight guidance to help information workers maintain focus on relevant windows, we aim to reduce multi-tasking activities that are distracting and unproductive. To make our investigations tractable, we focused our work on one community of information workers, software developers. We studied developers, because of their extensive use of computers at work, their openness towards improving their work and productivity [24], and overall comparable work, while working on a broad variety of activities [31, 9, 15]. Although the findings of our research are specific to software developers, we believe that our approach may have potential value for fostering focus and reducing distraction for other types of information work that entail multi-tasking.

To investigate the idea of reducing visual clutter to increase focus, we designed and developed the *WindowDimmer* application which predicts the most relevant windows currently in use and dims the other ones to reduce their likelihood of drawing the information worker's attention. To test the value of this approach, it was necessary to first establish a better understanding of developers' practices in interacting with conventional windows-based desktops at work, and specifically to understand their patterns of opening, closing, and switching between windows and tasks. We conducted a formative study to monitor computer and eye-gaze interactions of 18


*CHI '20, April 25–30, 2020, Honolulu, HI, USA.*

ACM ISBN 978-1-4503-6708-0/20/04.
https://dx.doi.org/10.1145/3313831.3376285

professional developers at work. The collected data led to surprising insights into the frequency and brevity of developers' window switching behavior (only 15.7 seconds spent per window), and also provided a baseline against which we could compare participants' window switching practices when using WindowDimmer. Our subsequent preliminary evaluation of WindowDimmer with 12 developers in the field indicated a substantial reduction in brief window switches and visits to windows irrelevant to the current task. Further, participant feedback to the application was generally positive, and indicated that WindowDimmer could help reduce distraction and improve focus. This paper makes these primary contributions:

- A monitoring study capturing information on window interaction practices of professional software developers.
- The WindowDimmer application, capable of predicting the relevance of all open windows and applying visual prominence to dim less relevant windows.
- An evaluation of the visual prominence approach with a second group of professional software developers and software engineering students.

## RELATED WORK

Related work can be roughly categorized into studies on understanding information workers' interactions with their window environment, approaches to detect currently relevant windows, and approaches to support task focused work.

### Window Interaction Studies

Modern window-based desktops allow information workers to access and see many applications and windows simultaneously with many applications capable of opening multiple windows. With increasing screen size, multi-monitor support, and the ability to use several applications simultaneously, they also provide sources for distractions, such as multi-tasking between the main and less relevant task, thus reducing focus [1]. Each window switch can potentially act as a trigger to decoy the worker to perform a task switch, such as a new unread email, an interesting Twitter conversation, or a not yet finished task. In addition to creating distracting visual clutter, work by McMains et al. [30], Jeuris et al [20] and Niemelä et al. [38] has shown that having more windows or tabs open on the computer at the same time, increases cognitive costs and requires more time for workers to find the relevant window. Little is known yet about information workers' practices in interacting with conventional windows-based desktops at work, such as opening, closing and switching between windows and tasks. Existing studies vary in the type of information that was tracked. Mostly, studies either focused on active observations [48] or specific computer interaction events based on keyboard and mouse interaction with specific applications [13, 16]. The interactions of software developers in particular have been studied within their IDEs for the Pharo IDE [35] and Eclipse [37]. Our work is not restricted on an individual application, but considers interactions with all programs and windows on a developers' computer.

In 2004, Hutchings et al. conducted a more generic monitoring study across all windows to examine the effect of larger screen real estate on window switching behavior [17], and found that users with smaller screens had a median of 4 visible windows, while those with larger or more screens had a median of 6. With ever increasing screen sizes and resolutions, we expect an even larger number of open and visible windows. Our studies also monitors this and provides an update to these numbers.

Further studies that observed information workers used the interaction to determine tasks [31] or higher-level working spheres [15, 26]. These studies were conducted over multiple days and reported a very fragmented working style with many task switches and interruptions.

In our first study, we leverage eye-tracking technology to gather task-independent insights into how software developers visually interact with the windows on their desktop. Eye-tracking has traditionally been used to study reading and comprehension. Early experiments studied the differences in code comprehension between novice and expert developers [5, 8]. Other research used eye-tracking technology to investigate the comprehension of specific software artifacts, namely class diagrams [56], design patterns [47], and identifier styles [46]. In more general contexts eye-tracking was used to measure task difficulty [4, 14], mental workload [18], and cognitive load [22], but not for the visual interaction with windows.

### Detecting Relevance

There is a range of approaches that have tried to detect the relevance of windows [39], artifacts [21], and groups of applications [41, 49]. Applications for using the relevance of related resources include task resumption, task switching, and self-monitoring. Most of these approaches take as basis two types of features: temporal and semantic. Temporal features are related to the order in which resources are accessed. For instance, Bernstein et al. developed a model based on the number of switches between two windows to rank their relatedness [6]. Semantic features are usually shared words in the window title and content, and were for example used to group open applications and documents into tasks [11]. Oliver et al. used both types of features to analyze window switches [40] and provide an alternative that reorders or highlights the windows in the window switching application [39].

### Supporting Focused Work

The high fragmentation of a software developer's typical workday is well reported [15, 26, 45]. On average, there are switches between different activities every few minutes even when not all of them are task switches [31]. Times with fewer task switches and higher engagement with tasks are reliably rated as more productive [29, 32]. Approaches to increase the perceived focus and productivity have, for instance, decreased distractions of work-unrelated websites by reducing their availability [53], or by fully blocking them [25].

There are several approaches to assist users with window switching [19, 50, 51, 55], but their work focuses solely on providing a better overview and speeding up window switching. Others, more closely related to our work, have tried to reduce the overload that people experience by grouping tasks and documents [11], grouping windows [49], improving access to occluded window content [54], or reducing the visibility of secondary screens [10]. The latter approach by Dostal et al.

tested several approaches on how to reduce distraction from a separate, non-focused screen. Their best approach used a dimming of the screen while highlighting display changes.

Specifically for software developers, approaches to reduce the "window plague" in the window-based Pharo IDE use temporal features to highlight the most important windows and automatically close the least important ones [36, 42]. Both of these approaches are restricted to the IDE (a single application) and were never tested with users.

## STUDY 1: DEVELOPERS' WINDOW INTERACTIONS

To support developers' focus at work by emphasizing relevant windows, a first step was to establish a better understanding of their current practices in interacting with conventional windows-based computer desktops. In particular, we studied developers' patterns of opening, closing, arranging and switching between windows on their computer monitor(s).

### Monitoring Study

In this first formative study, we monitored 18 professional software developers' window interactions for an average of 17 days (4 to 37, ±10.2) per participant in their usual work environment. During the study, we ran a monitoring application on participants' computers to collect window interaction and user input data, and collected eye-gaze data via an eye-tracker.

#### *Participants*

We recruited 18 professional software developers, 1 female and 17 male, from three companies of varying size in the software and computer hardware industry through personal contacts. We limited our study to users of Microsoft Windows as their main operating system due to the compatibility with the eye-tracker and our monitoring application. At the time of the study, participants had an average of 17.5 years of professional development experience, ranging from 2 to 35 and described their main responsibility as development with accompanying tasks of project management, system engineering, and testing. They all resided in Switzerland.

#### *Procedure*

At the beginning of the study, we provided participants with detailed information on the study procedure and goals, the data we were going to collect with the monitoring application and eye-tracking sensor, and asked them to sign a consent form (approved by UBC's Research Ethics Board). We also informed participants that their participation is completely voluntary and without compensation and that they are allowed to withdraw at any point in time. We then asked and supported participants with installing the monitoring application and the proprietary eye-tracking software on their computer, and with correctly positioning the eye-tracker in front of their primary monitor. We used the Tobii 4C eye-tracker [52] due to its portability, affordability and reliable results in previous work. We only captured eye-gaze data on participants' primary monitor, due to technical limitations that restrict the use to one Tobii eye-tracker per computer. After calibrating the eye-tracker with participants, they were asked to continue their regular work as usual, while the monitoring application and eye-tracker were running non-intrusively in the background. To ensure a high reliability of the eye-gaze data, Tobii 4C's software constantly reassesses whether the sensor needs to be re-calibrated based on the head movement, and if necessary, prompts the participant to do so.

At the end of the study, we revisited each participant, collected the monitoring data from the participant's device, uninstalled the software, removed the eye-tracker, and conducted a short follow-up interview to ask for feedback on the study.

#### *Monitoring Application*

To collect computer interaction and eye-tracking data, we used our own monitoring application, PersonalAnalytics [34], that we developed and used in previous studies (e.g. [33, 31]). Our application collects participants' user input data, including mouse movement and keyboard events, without logging the specific keys for privacy reasons. In addition, the application logs all window events, including focus, create, destroy, move, and resize events together with the size and location of the window, the title of the window and its state (active, minimized or maximized), information on all other open windows and their position, as well as on all connected screens and their dimensions. To collect eye-gaze data, our application captures eye-fixations—the location on the screen the user visually pays attention to—including the fixation position and duration, by accessing the data from the Tobii 4C's proprietary software.

#### *Collected Data*

We collected a total of 322 days of data for this study, ranging from one to four weeks worth of data per individual participant. All participants reported the days they were monitored on as regular work and did not report any distraction or changes in their work based on our monitoring application and the eye-tracker. Overall, we collected a total of 9,522,956 window interactions and 63,292,622 eye fixations from all 18 participants. From the eye-gaze data, we discarded all data that was not accurate according to Tobii's quality rating.

For each window interaction, we calculated how many pixels of each open window was visible based on the order in which windows appear on the monitor. We further determined how long windows were open or active based on the duration between window events. From the mouse movement data, we derived periods of time when participants were not active on their computer and excluded these from our analysis. In particular, we considered every period of 5 or more minutes without mouse movement as inactive and excluded it.

To identify the windows a participant looked at, we mapped each captured eye fixation to a window using our processed window interaction data that includes the visibility of all windows, their location and size. Overall, we were able to distinctively map 86.6% of the recorded fixation points to a window. In cases when a participant looked at the taskbar, a monitor with no windows, or a newly opened window that was not yet registered with the windowing system (which can happen in the first second of opening a new application), we were not able to map the eye fixation to a window.

By comparing and validating the data of all participants, we noticed one outlier, participant P6, that had more than twice the amount of open windows compared to all other participants across the three companies. Since our objective was to

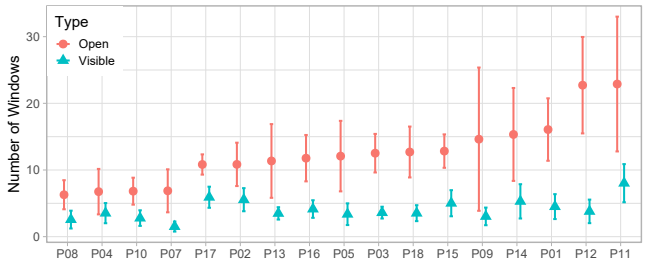


**Figure 1. Average number of open and visible windows by participant.**

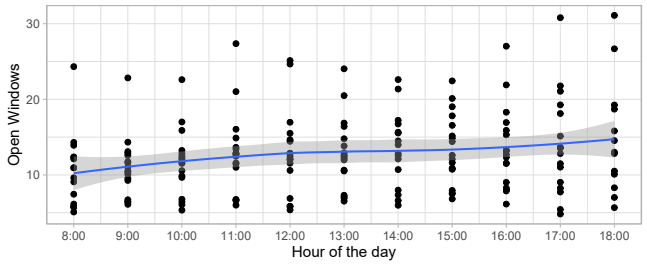


**Figure 2. Weighted mean of open windows by participant and hour during a typical work day where we could gather enough data from all participants. The line shows a local regression (Loess).**

gather a better understanding of the usual developer practices in interacting with windows-based desktops, we decided to exclude the data of P6 from the further analysis.

## Results

This section presents the results of our quantitative analysis of the logged computer interaction and eye fixation data.

### *Open Windows Behavior*

To better understand the potential of window switching to distract developers, we analyzed how many windows and applications developers have open at any given time. We calculated the weighted number of open applications and windows by the duration they were open. Figure 1 presents the number of open windows across participants. Overall, developers had an **average number of 12.1 (±6.6) windows open at all times** from running an average of 9.5 (±3.0) applications. When considering multi-monitor setups, we observed that developers have more windows open when using more monitors: single-monitor users tend to keep an average of 6.9 (1 participant only), dual-monitor users 12.6 (±4.0), and triple-monitor users 14.9 (±11.4) windows open at the same time, which is comparable to previous work [41]. The overall screen size, however, did not significantly impact the number of open windows (Pearson correlation coefficient of 0.21, p=0.42).

We found that the number of open windows varies over the course of a day. In particular, developers opened more windows than they closed, leading to a **growing number of open windows over the course of the workday**, from 9.9 (±5.5) in the morning to 14.4 (±7.3) in the evening. Despite the variation in the number of open windows, this increase over the course of a day is consistent across *all* participants as illustrated in Figure 2, and similar to the increase in the number of windows open inside the Integrated Development Environment (IDE) that Roethlisberger et al. found [42]. Developers

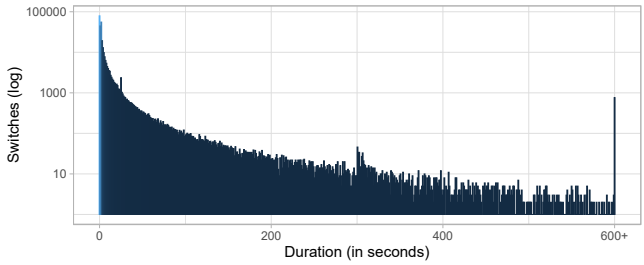


**Figure 3. Number of window switches by time between window switches in seconds. The bump at 300 is caused by our 5 minute idle timeout.**

generally do not close windows frequently, but rather leave them open in the background, although, some developers at least tend to reorganize and clean up their desktop a bit more after returning from lunch break or before leaving work.

We also found that developers very rarely move or resize windows, compared to the frequency of other window interactions, such as switching, opening and closing. Less than 3% of all window interaction events that were captured over the course of the study were changes of size or location of a window.

### *Window Switching Behavior*

Based on the computer interaction data, we found that each window was open for a total of 79.2 (±60.1) minutes on average. Yet, this value varied a lot, ranging from 19 minutes to 4 hours across participants. Nonetheless, we found that developers **switch between windows very frequently keeping a window in focus (active) for only very short amounts of time**. On average, developers focused on a specific window for only 15.7 (±77.3, median=2.2) seconds, before switching to the next one. The high number of switches is comparable to previous work [15, 31, 33] that looked into activity and task switching, which are related to window switching.

As illustrated in Figure 3, the short time developers spend in a window is right skewed by a high number of very short window switches. Overall, 32.9% of all window switches lasted less than 1 second and only 30.7% of the windows remained active for longer than 5 seconds before the developer switched away. The large number of short window switches might occur for several reasons, including the developer navigating through the open windows to find the relevant one, the developer closing irrelevant windows or selecting the wrong one, or from being briefly distracted. Most window switches are switches to previously open windows. In total, developers revisited already open windows in 76.6% of the window switches and only in the remaining 23.4% they opened a new window.

### *Multi-Monitor Usage*

Even though 94.1% (16 out of 17 participants) are using a multi-monitor setup, our data shows that developers are using their **primary monitor during the majority of the time** they spend on the computer and position the most windows on it. On the Windows 10 operating system, one monitor is defined as the primary one, the remaining ones are secondary monitors. The secondary or tertiary monitors were actively[1] used only

[1] An "active window" means it is currently focused and/or receives keyboard input.

during 30.7% (±23.3%) of the time, and for 23.9% (±19.8%) of the windows, on average. 14 participants (82.4%) worked with two monitors, 2 participants (11.8%) worked with three. Several developers used a laptop and connected the laptop to an external monitor for most of the time, but sometimes also just used the laptop and therefore only a single monitor configuration. Over the course of the study, we found that 64.7% (11 participants) switched their monitor configuration from time to time, while 35.3% (6 participants) consistently used the same monitor setup throughout the study. Overall, developers switch windows more often on the primary screen (32.3% compared to 26.8%, two-sided paired t-test p=0.05276) and have each window open for a shorter amount of time (67.9 ±47.6 vs 108.2 ±81.6 minutes, p=0.05882). The average duration a window remains active, before being switched away, is the same on all monitors, with 15.7 seconds.

#### *Usage of Screen Real Estate*

We found that developers have **several windows open and visible at the same time**. Over all monitors, there were always an average of 1.5 (±0.7) windows fully and 3.3 (±1.7) windows partially[2] visible. The currently active window, which on Windows 10 is always fully visible, thereby took up 84.7% (±9.7%) of the monitors' screen real estate and was maximized[3] 60.6% (±48.9%) of the time.

#### *Visual Attention and Focus on Windows*

The results on developers' window switching behavior so far are based on developers' computer interaction. Hence, it is still unclear whether developers are actually looking at the active window, or at another window that is currently not in focus. Since a better understanding of the visual attention also allows us to better understand which windows are relevant, we analyzed the eye-tracking data from the' primary monitors.

The results of our analysis show that **visual attention shifts similarly frequently as the active window**, with an average of 31.2 (±87.8) seconds per window and a median of 4.7 (±20.4) seconds. The distribution of shifts in visual attention between windows is again heavily skewed, with 19.3% (±8.4%) of the shifts being less than a second long. The longest time we observed a developer looking at the same window was 64 minutes, significantly lower than the 4.7 hours for the active window. This is however not surprising, since developers might just look away in between for short periods to relax their eyes.

Using the captured eye fixations, we further found that the **visual attention matches the actively selected window in 83.1% of the cases** on the primary monitor. The other 16.9% of the eye fixations were directed at windows that were open and visible, yet not the active one. It is not surprising that developers do not always look at the active window, and in fact, developers looked away from the active window at least once for 67.7% of all active windows. When they did so, they looked at an average of 2.2 (±3.2) distinct non-active windows. There were, however, also 32.3% of the active windows that participants never looked away from until they switched the active window using mouse or keyboard. Overall, this analysis provides evidence that using the active window is a relatively good indicator for a developer's attention.

[2]In a partially visible window, another window is partly overlapping, and thus covering, it.

[3]A maximized window takes up the entire screen real estate of the monitor, except the taskbar.

### Discussion

The first study revealed that most developers keep a large number of windows open at the same time, which could lead to distractions that result in switching to a non-relevant window as previous research has shown [1, 30, 38]. The high frequency of window switches and short duration a selected window stays active further suggests that many of these switches might be non-strategic or unintentional switches that could distract from the primary task.

We saw no evidence that developers took any explicit, systematic measures to reduce distraction from less relevant windows. Our findings and discussions with participants, however, suggest that there is potential value of a lightweight automated approach to reduce distractions by open windows as a means of supporting developers in their focused work.

Additionally, our results suggest that the computer interaction can be a good indicator for determining a developer's visual attention based on the 83.1% overlap for the primary monitor. Therefore, we rely on this interaction as the basis of our relevance model due to the potential of its applicability and low invasiveness for workers.

## PREDICTING RELEVANT WINDOWS

To develop a lightweight approach that helps software developers maintain focus on the relevant windows and minimizes unintentional switches, we explored the prediction of relevant windows.

### Model

We leveraged the results from study 1 to develop a model on predicting the most relevant windows. We used the following temporal and semantic features that have been applied successfully in previous work [11, 40, 39]. Note that other features that we identified from related work or our own experience would have made the model either too complex or would not allow a real-time application in a real-world setting that we implemented in study 2.

#### *Temporal, Recency*

Recency is one of the simplest and most commonly used temporal features that scores a window based on how recently it has been active, so that the last window before the current one receives the highest score. To calculate a recency score we focus on the window activation events in the computer interaction event log. Specifically, we start from the current window and go back in time to the ten previous windows, remove duplicates, and give a score in reverse chronological order. From an order of $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D \rightarrow C$ with $C$ being the active window we get a ranking of $D, B, A$ as the active window is not scored and duplicates are removed.

#### *Temporal, Duration*

Our duration feature scores a window by how long it has been active during the last 10 minutes. The score is calculated

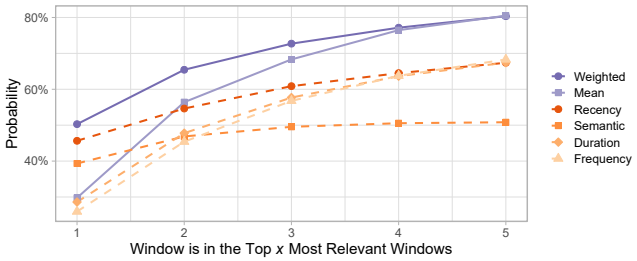


**Figure 4. Probability of predicting the correct next window within the top $X$ most relevant windows.**

by dividing the summed active duration by the observation interval, in our case 10 minutes. We picked this interval to provide a good balance between keeping the measure stable from short recent switches and only including windows that are related to the current task. Task switches are reported to occur every 3 to 12 minutes, so the current task should in most cases be entirely covered by our interval [29, 31].

*Temporal, Frequency*

The frequency feature shares the same observation interval with the duration feature. Windows are scored by counting the number of switches to that window: the more switches to a window, the higher the rank of the window. Many tasks involve switching frequently between the same few windows. This feature is intended to capture that behavior.

*Semantic, Window Title*

We assume that windows whose title and textual content are similar are related and therefore relevant to each other. We want to use semantic features to predict how related two windows are. To calculate a score in regard to predicting window switches we are interested in how related a window is to the currently active window. We are using the window titles to determine relatedness. While the contents of a window could provide more information, they are not as easily available as the window titles, which are exposed by operating system interfaces, and reading the contents of all windows is very invasive, introducing justified privacy concerns by users. We processed the window titles by removing all punctuation, filtering out stop words, and stemming the remaining words. The weight of the stemmed words in the processed window titles of all open windows are calculated by the term frequency–inverse document frequency. All processing operations are performed with *tm*, a library for text mining in R [12]. With these weights we calculate the cosine similarity of the title of each open window to the title of the currently open window. The open windows are ranked by the similarity value.

### Empirical Analysis based on Study 1

After calculating the scores for the temporal and semantic features described above, we used the data collected during our monitoring study to determine the weight of each of the features to calculate a combined score. As a starting point, we used a linear equation due to its simplicity and speed of calculation. For each window switch, we calculated a score for each open window equal to $\alpha * recency + \beta * duration + \gamma * frequency + \delta * semantic$.

We investigated the features individually and created a ranking based on their predictive power. Figure 4 shows that the order is not easy to determine and depends on the target number of windows that are predicted. When we consider only our highest scoring window, the feature with the highest predictive power is recency, followed by semantic, duration, and then frequency. When considering the top 2 or top 3 results, the order changes as the semantic feature becomes less predictive.

We tested all features with equal weight as well as all combinations of 3 out of 4 and 2 out of 4 features. Figure 4 shows the performance of the four individual features, a combination using equal weights (*mean*), and our final weighted combination (*weighted*). Using equal weights performs significantly worse than either of our best two features. The best combination we found only consisted of three of our features containing recency, semantic, and duration. Adding frequency reduced the result slightly. We then proceeded to examine various weight schemes of the three taking into account the order. We used a binary search approach with intervals of 0.2, 0.1, and finally 0.05 to gradually optimize the weights. The optimal combination we found that way is $0.5 * recency + 0.45 * semantic + 0.05 * duration$. While its weight ended up being rather small, the duration feature improves the score by filtering out switches to windows that become active and relevant for a very short time, but should not change the whole context of the task.

## STUDY 2: DIMMING LESS RELEVANT WINDOWS

We explored whether *visual prominence* can be leveraged to reduce clutter and distraction. To that purpose, we extended our monitoring application with a component, called *WindowDimmer*, which de-emphasizes less relevant windows by fading them away, and thus, emphasizing relevant ones.

### WindowDimmer Approach

Using an iterative design process, we developed several versions of WindowDimmer, including masking all but the relevant windows. We piloted the iterations with four students at our universities, and eventually settled on an intervention that was lightweight enough to be used in real-world work, without creating any additional distractions or requiring reorientation. The final version extends the monitoring application that we developed for study 1 with the model to predict relevant windows that we determined in the empirical analysis. WindowDimmer emphasizes the 3 most relevant windows and reduces the visibility of all others. We chose 3 as the threshold since we learnt from analyzing developers' eye-gazes that often times, not only the active window is fixated but an average of 3.2 distinct windows including the active one. Hence, we predict a ranked list of the 3 most relevant windows using our combined and weighted predictor (*weighted* in Figure 4) which reaches a 72.7% probability of selecting the correct three most relevant windows for the study 1 data. Predictions by a random model are correct significantly less often with a probability of 33.9% (Monte Carlo simulation, p<2.2E-16). Whenever WindowDimmer receives an event that the user switched from one window to the next, we gather a list of the remaining open windows and their window titles, extract the features, and calculate the relevance score that we described

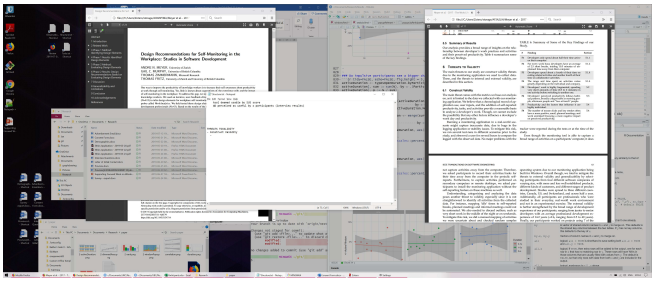

**Figure 5. WindowDimmer dims all less relevant windows and the desktop to emphasize the top 3 most relevant windows.**

above. Whenever a window title changes without a window switch, such as when the user selects a new tab inside the web browser, we also recalculate the relevance scores.

As visualized in an example in Figure 5, WindowDimmer reduces the visibility of all windows that are not the top three most relevant ones, by dimming them. Concretely, the active window and the two windows with the highest relevance scores remain untouched, while all other remaining windows and the desktop background, in case it is still visible, get slightly dimmed by applying a gray overlay with 25% opacity. The Windows 10 taskbar stays visible and un-dimmed to allow easy navigation between windows. Note that the most relevant windows are not necessarily all in the foreground, since the order of how Windows 10 stacks the windows is based on recency only, while our model includes duration and semantic similarity as well. Finally, WindowDimmer allowed participants to turn off or pause the approach, and to adjust the number of relevant windows or the dimming.

## Intervention Study

To evaluate the potential of reducing clutter and increasing focus by dimming less relevant windows, we conducted a second, preliminary field study in a real-world-setting. In the first part, we investigated if our model can identify the top 3 relevant windows as reported by participants. In the second part, participants used WindowDimmer *in situ* during their real-world work and provided us with qualitative feedback.

### *Procedure*

In the beginning, we asked participants to read the consent form (approved by UBC's Research Ethics Board), ask any questions, and sign it. Participants were then asked to install the monitoring tool with the WindowDimmer extension on their main work computer. Finally, we asked participants to continue their work as usual for the next 5 workdays.

During the *first* part of study 2, which lasted three of the five days, we examined whether our predictive model is able to accurately predict user-defined relevance. To that purpose, we logged participants' interactions with their windows and asked them to answer a pop-up survey in regular 40-50 minute intervals (to have some randomization). The pop-ups listed all currently open windows and asked participants to tick a check box of all the windows that are relevant for their current work. No definition for relevance was given, as we did not want to bias participants' responses by restricting relevance to work-related windows for example.

During the *second* part of study 2, which lasted the remaining two days, we continued logging participants' computer interaction and enabled WindowDimmer. We disabled the self-reporting pop-up to prevent frustrating participants in case their selection of relevant windows was different to the windows that our approach dimmed. At the end of the second part of study 2, we conducted semi-structured interviews to receive feedback on their experience with WindowDimmer. We also asked them to answer a survey with demographic questions as well as the System Usability Scale [7], a standardized survey for evaluating the usability of our approach. Finally, we collected the logged data from participants' machines, after giving them the opportunity to obfuscate the logged data to alleviate potential privacy concerns. Finally, we uninstalled the monitoring application and gave participants $30 US to compensate for their efforts.

### *Participants*

We recruited a total of 12 software developers, 5 female and 7 male. 6 are professional software developers who worked for 4 different software companies in Canada, the US, Germany, and Switzerland, and 6 are computer science students (1 undergraduate, 4 graduate, 1 postdoc) in Canada and Switzerland. We recruited participants through personal contacts. At the time of the study, participants were on average 26 years old. The 6 participants who identified as professional software developers have an average of 2.6 years of experience. Participation was entirely voluntary.

### *Collected Data*

To evaluate our approach, we collected data on connected screens, open windows, and window interaction comparable to the monitoring study. We additionally recorded the calculated relevance scores for each feature as well as the summed score and rank of the open windows. For each window in a submission of the pop-up by a participant, we recorded the response and whether our model predicted the window as relevant or would have dimmed it. In total we collected 266 pop-up responses with an average of 22.2 (±9.5) responses per participant over three days. The number of responses varied between 11 and 36, depending on how much time the participant spent on their computer. Each pop-up contained a list of 14.5 (±11.8) open windows.

## Results

The intervention study provides similar data to the monitoring study, but extends it with information related to our model and the WindowDimmer approach. We compare data on computer desktops, windows, and window interaction with the previous results, investigate participants' reports on relevant windows, and compare window interaction behavior between the two parts of the study.

### *Window Interaction Behavior*

The participants of the intervention study had comparable screen setups as the participants of the previous monitoring study. All participants used a laptop as their main machine with six participants adding one and three participants adding two external monitors. We calculated the number of open windows with the same methodology as before and observed

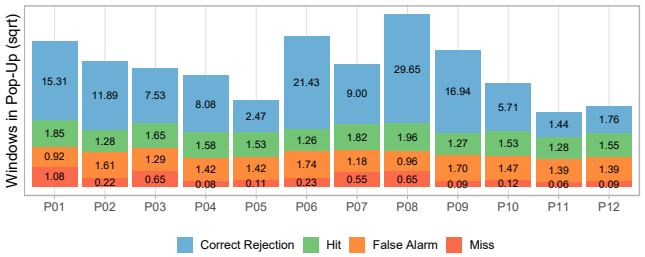


**Figure 6. Number of windows reported (by participants) and predicted (by our model) relevant averaged over all reports. The three top scoring windows in our model are considered relevant.**

13.5 (±11.1) open windows across participants, ranging from 4 to 31, which is a slightly higher average compared to the 12.3 open windows in the monitoring study (study 1).

Only 17% of the captured window switches lasted less than a single second, compared to 30% in study 1, but with a similar distribution as seen in Figure 3. The proportion of window switches to already open windows is 67.8%, also slightly lower than what we found in study 1 with 76.6%. These differences could be explained by the different applications used, tasks pursued or participants' job roles.

*Predicting Relevant Windows*

To evaluate our model's performance in predicting relevant windows, we compared our model's predictions to the participants' reports on the relevance of the open windows. In the 266 self-reports that we collected in the first part of the study, **participants reported 1.8 (±0.9) windows to be relevant out of the 14.5 (±11.8) windows that were open and listed in our pop-up.**

As described before, we consider the top 3 scoring windows in our model as relevant. Our approach was able to **determine the relevance correctly for 88.3% of the windows**, were windows matched the self-reports (10.4% hits, 77.9% correct rejections). 9.8% of all windows were predicted relevant by our model, but not by the participants (false alarms). Only 1.8% should have been considered relevant according to our participants, but were not predicted as relevant based on our model (misses). Compared to a random model, the accuracy improved significantly from 60.8% (Monte Carlo simulation) to 88.3% (our model) with p<2.2e-16, and the balanced accuracy improved from 57.7% to 86.8% (p<2.2e-16).

Figure 6 shows the relevance results per participant. To account for the varying number of responses per participant, the values are averaged per pop-up. The number of windows incorrectly predicted as not relevant, incorrectly predicted as relevant, and correctly predicted as relevant are very similar across participants. The value that is varying the most is the number of windows that were predicted and reported as not relevant. This is most likely due to the the varying number of open windows per participant.

*Evaluation of WindowDimmer*

During the study period in which participants used WindowDimmer an **average of 30.4% of the computer desktop area (all screens) was dimmed**. The results also show that


**Figure 7. Distribution of the duration a window is active before and after the dimming is activated by type of application. The distribution is binned into 3 buckets (0-1s, 1-5s, 5+s) to highlight the changes.**

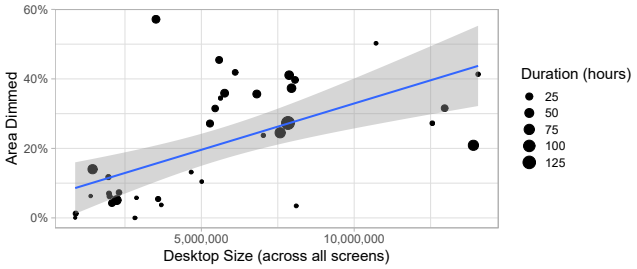


**Figure 8. Percentage of desktop area dimmed by desktop size. Both are calculated across all screens. The line shows a linear regression (`lm`).**

the dimmed area increases with the size of the computer desktop with a Pearson correlation of 0.59 (p=0.00011). Figure 8 illustrates this effect. During the study, all participants used a laptop with 9 of the 12 participants usually connecting at least one additional screen.

During the intervention phase with WindowDimmer activated, participants had **fewer window switches and spent longer time in relevant windows**. The number of window switches lasting less than one second decreased from 18.4% to 15.3%. However a paired t-test showed that this is not significant (p=0.1372). Figure 7 displays a breakdown by application type of the length of window activations. While there is a decrease of very short window activations across all types of applications, the ratio of very short window interactions varies. Applications that require a higher level of focus and concentration like IDEs and web browsers stay active for longer times while email clients have more very short activations.

Having WindowDimmer reduce visibility of windows might encourage participants to open more windows as they are no longer as distracting. Yet, we do not see a consistent change in the number of open windows across participants. While the number decreased for 6 participants, it increased for the other 6. The number of windows seems to be much more related to the type of work and tasks performed during the day.

For each window switch by a participant we recorded whether our model predicted the target window to be relevant and the rank of the target window. Window switches to windows not predicted to be relevant and therefore dimmed during the second part of the study decreased by 10.2%, from 48.2% to 43.3% with the dimming active while switches to the window WindowDimmer predicted as most relevant increased from 29.8% to 33.1%. Both changes are not significant with p = 0.9975 and p = 0.8383 respectively. The remaining switches target the second most relevant window, which is the third window not to be dimmed, where we also saw an increase by one percentage point.

### Participant Feedback

Our post-study survey included the System Usability Scale (SUS). We measured a mean SUS score of 72.7 which is considered to represent a "Good Usability" based on a large survey of SUS scores of previous studies [3].

We further interviewed participants after they completed the study to collect qualitative feedback and ask about specific situations, where they perceived our approach to be helpful or hindering to their work. Generally, the WindowDimmer approach was perceived as useful by 8 of the 12 participants. They mentioned the window dimming was "helpful", or "worked well" for them. The other 4 participants reported it having no or very little effect, but also not hindering their work. No participants stopped using the tool, which suggests the lightweight approach was not perceived as too distracting.

*P9* and *P11* reported an interest in dimming everything but the currently active window "to really focus on the current task". Leaving only one active window undimmed instead of three might help them to stay better focused. Additionally, they liked that the dimming not only applied to other windows, but also the desktop background itself. These two participants found their desktop background was "normally very cluttered and that can be pretty distracting". With one participant calling himself "not a great organizer of my desktop", dimming the desktop background reduces the focus on the clutter.

Three participants (*P6*, *P9*, and *P10*) found themselves distracted by sudden changes in the dimming. When they had too many open windows, they wanted to make sure WindowDimmer was not dimming anything important, which caused them to look at windows that just had the dimming applied. *P9* and *P10* would prefer a smoother transition leading to the full dimming over a few seconds.

All participants generally agreed on the problem of decreased focus when working on their window-based computer desktops. Although outside of the scope of WindowDimmer, many participants reported having trouble finding the relevant content within tabbed interfaces, especially the web browser. Two participants (*P2* and *P10*) suggested applying a similar dimming approach to tabs they hadn't used in a while and were no longer relevant.

## THREATS TO VALIDITY

The biggest threats to the validity for our results are external validity threats due to the limited number of participants and the focus on studying professional software developers as one type of information workers only. We elaborate those and the threats to construct and internal validity in this section.

### Construct Validity

Both studies were conducted in the real world to gather data that is as realistic as possible. Using a monitoring application in this unsupervised scenario bears the risk of causing inaccurate data to be included due to bugs in the implementation or unexpected restrictions of the participants' device. To mitigate this risk, we built our monitoring application by extending an existing application that we used in previous studies [33, 31] and conducted study test-runs on various machines prior to running both studies.

Limitations to the logged data are that the monitoring application can only capture the participants' interactions with the computer, and not away from it. Also, the eye-tracker did not allow tracking multiple monitors at the same time, which is why our eye-gaze data is limited to the primary screen only.

### Internal Validity

Monitoring participants might implicitly influence participants' behaviors, since they are feeling observed. In order to mitigate this risk, we constructed our monitoring application in a way such that data can be collected in the background only. Many participants reported that they forgot about the monitoring application shortly after installing it. In study 1, the eye-tracker could have reminded participants of the ongoing study, but no other interaction was required. In the first three days of study 2, the periodic self-reports (pop-up to collect window relevancy data) might have been considered more intrusive, but participants did not state anything with that regards in the post-study interview. To reduce the intrusiveness of the pop-up, we minimized the amount of time required to select the most relevant windows by adding application icons for quick identification and only requiring a single click to select each relevant window. We further allowed participants to skip individual pop-ups to prevent them from submitting inaccurate information in situations where they were unable to spend enough time selecting the actually relevant windows. In the second part of study 2, where we applied WindowDimmer, we actively influenced participants' window switching behavior, but this was the intention of this intervention. Since the evaluation of WindowDimmer is based on limited data from two days, we cannot exclude that the positivity towards the usefulness of WindowDimmer was caused by novelty effects. While our study showed there is potential, running the field-study over several weeks and with more participants in the future would help to more clearly demonstrate its value.

### External Validity

Our selection of participants and the total number of participants for either study could limit the generalizability of our findings. While all participants were information workers, participants in study 1 were professional software developers, 6 (of the 12) participants in study 2 were computer science students. We believe that recruiting software developers as a type of information worker for the first study is a good starting point, also to be able to compare better between individuals. Overall, we further tried to mitigate the threat to generalizability by recruiting participants from different companies, countries, and contexts. Further research is needed to validate our approach with a broader set of information workers.

The architecture of our monitoring application required us to restrict the study to participants using Microsoft Windows 10 as operating system. Further studies with extended tooling are required to assess the effect the window managers of different operating systems have on the window interactions and focus.

## DISCUSSION AND FUTURE WORK

The effects of the use of WindowDimmer on software developers' window switching patterns and the participants response to the tool suggest that it helps to foster greater focus in work

tasks and also reduces the distraction of irrelevant windows. Using WindowDimmer, participants engaged longer with relevant windows and switched windows less frequently. We believe this effect can be attributed to the system's ability to predict window relevance with relatively high accuracy and increase the visual prominence of relevant windows in relation to irrelevant ones.

Despite the benefits that WindowDimmer offers, there are several important issues that arose in our study that need to be addressed going forward. In particular, although the relevance model was able to predict window relevance with high accuracy, the frequency of false positives is not trivial, and these windows when not dimmed add unnecessary visual clutter. The false negatives, while very few, are also a concern as the dimming of a necessary window could result in additional time and cognitive burden to locate, potentially reducing the person's focus on the main task. For these reasons, further improvement of the relevance model is a priority going forward. Our model for predicting relevance uses three temporal and semantic features and a linear combination with heuristic weighting of these features. Building on a larger dataset, adding more features, and using machine learning techniques could improve our model significantly. Our studies revealed large differences in the window interaction behavior between our participants. To achieve the best relevance prediction, a personalized model might be required. Some participants wished for the functionality of overriding the relevance by manually setting the dimming of a window. A more sophisticated, personal model could utilize these inputs.

Although the overall task focus improved, as indicated by the window switching behavior, it is also still a concern that the window dimming feature could be a distraction itself. Participants mentioned that the dimming feature drew some attention as they wanted to be certain relevant windows were not being dimmed; this effect is clearly undesirable and could break the worker's focus on the primary task. Therefore the design of the interaction should be considered further, including the possibility of more gradual animations, staggered dimming of windows over a longer period of time, or subtle fading effects that reduce the visual prominence of irrelvant windows without substantially darkening the windows. The approach of predicting and prioritizing relevant windows could also be extended to handle more complex environments as well. Similar to previous studies [17], we have seen the size of monitors and number of windows increase. This trend is most likely to continue, increasing the potential for visual clutter from open windows. We have focused on increasing the visibility of relevant windows as a simple yet effective first step, but can imagine approaches with a larger impact like minimizing or hiding windows that are not relevant and grouping related windows that are related to a task to have faster access.

Additionally, it may be worth considering increasing the user agency and interactivity of WindowDimmer, as well as making use of context to optimize the functionality. Our approach features a model that runs in the background and passively collects user data automatically. To adapt to the various activities a developer engages in over the course of the workday, a more interactive approach could take user settings into account. A very specialized activity like programming could feature a lower number of not dimmed windows and a more change-resistant model, whereas a broader research activity could dim fewer windows and value the semantic relatedness of windows higher. Even more direct control over dimmed windows could give users the ability to toggle whether a specific window is dimmed or manually mark windows relevant or not relevant for the current task. The individual labels could further be used to dynamically improve the relevance model.

Despite the many technical and interaction improvements that could be realized going forward, we believe that the results of the study provide evidence that the overall approach of predicting relevant windows and increasing their relative visual prominence is effective for reducing distraction and increasing focus on software developer's primary task. Pursuing the open issues described here could help optimize the approach further, potentially reducing distraction and improving focus to an even greater degree.

## CONCLUSION

With the constant improvements in hardware and software technologies, screen sizes and resolutions are increasing and usage behavior for window-based desktops are changing. Our monitoring study provides recent insights into how software developers setup their desktop environment and how they interact with windows on their computer. We observed their workday to be very flexible with a changing number of monitors, multiple open windows at the same time, and very frequent window switches. This mirrors the assessments of a fragmented workday found in previous work.

Based on the collected data, we devised a model to predict the relevance of open windows that showed an accuracy of 72.7% when predicting the 3 most relevant windows. Evaluating the model in a second study with a different set of participants in situ, showed that the model can predict self-reported relevance by participants for 88.3% of the windows.

To allow the model to be applied in a real-world scenario, we developed an approach called WindowDimmer, which reduces the visibility of windows that we predict to be less relevant. The results of our second study, a preliminary evaluation of the approach, in which participants were asked to use the dimming approach showed a reduction in window switches and an increase in switches to relevant windows when the dimming is active. The majority of our participants felt that WindowDimmer was useful in reducing clutter and supporting continuous focus on relevant windows. In the future, we plan to improve the usefulness and generalizability of WindowDimmer by training our model with more data, testing different visualization techniques, and evaluating it over longer periods and with other information workers.

## ACKNOWLEDGMENTS

We are grateful for funding from ABB's Industrial Research Grant F17-05273-22R7735 and Canada's NSERC CRDPJ 530226-18. We thank all our participants, and the anonymous reviewers for their extremely helpful feedback.

## REFERENCES


[1] Brian P. Bailey and Joseph A. Konstan. 2006. On the Need for Attention-Aware Systems: Measuring Effects of Interruption on Task Performance, Error Rate, and Affective State. *Computers in Human Behavior* 22, 4 (July 2006), 685–708. DOI: http://dx.doi.org/10.1016/j.chb.2005.12.009

[2] Brian P Bailey, Joseph A Konstan, and John V Carlis. 2001. The Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface. In *Interact*, Vol. 1. IOS Press, Tokyo, Japan, 593–601.

[3] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction* 24, 6 (July 2008), 574–594. DOI: http://dx.doi.org/10.1080/10447310802205776

[4] Jackson Beatty. 1982. Task-Evoked Pupillary Responses, Processing Load, and the Structure of Processing Resources. *Psychological Bulletin* 91, 2 (1982), 276–292. DOI: http://dx.doi.org/10.1037/0033-2909.91.2.276

[5] Roman Bednarik. 2012. Expertise-Dependent Visual Attention Strategies Develop over Time during Debugging with Multiple Code Representations. *International Journal of Human-Computer Studies* 70, 2 (Feb. 2012), 143–155. DOI: http://dx.doi.org/10.1016/j.ijhcs.2011.09.003

[6] Michael S. Bernstein, Jeff Shrager, and Terry Winograd. 2008. Taskposé: Exploring Fluid Boundaries in an Associative Window Visualization. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology - UIST '08*. ACM Press, Monterey, CA, USA, 231. DOI: http://dx.doi.org/10.1145/1449715.1449753

[7] John Brooke. 1996. SUS: A 'quick and Dirty' Usability Scale. In *Usability Evaluation in Industry*. CRC Press, London, 189–194. DOI: http://dx.doi.org/10.1201/9781498710411

[8] M.E. Crosby and J. Stelovsky. 1990. How Do We Read Algorithms? A Case Study. *Computer* 23, 1 (Jan. 1990), 25–35. DOI:http://dx.doi.org/10.1109/2.48797

[9] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A Diary Study of Task Switching and Interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, ACM, New York, NY, USA, 175–182. DOI: http://dx.doi.org/10.1145/985692.985715

[10] Jakub Dostal, Per Ola Kristensson, and Aaron Quigley. 2013. Subtle Gaze-Dependent Techniques for Visualising Display Changes in Multi-Display Environments. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces - IUI '13*. ACM Press, Santa Monica, California, USA, 137. DOI: http://dx.doi.org/10.1145/2449396.2449416

[11] Anton N. Dragunov, Thomas G. Dietterich, Kevin Johnsrude, Matthew McLaughlin, Lida Li, and Jonathan L. Herlocker. 2005. TaskTracer: A Desktop Environment to Support Multi-Tasking Knowledge Workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces - IUI '05*. ACM Press, San Diego, California, USA, 75. DOI: http://dx.doi.org/10.1145/1040830.1040855

[12] Ingo Feinerer, Kurt Hornik, and David Meyer. 2008. Text Mining Infrastructure in R. *Journal of Statistical Software* 25, 5 (2008). DOI: http://dx.doi.org/10.18637/jss.v025.i05

[13] K.D. Fenstermacher and M. Ginsburg. 2002. A Lightweight Framework for Cross-Application User Monitoring. *Computer* 35, 3 (March 2002), 51–59. DOI: http://dx.doi.org/10.1109/2.989930

[14] Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, Hyderabad, India, 402–413. DOI: http://dx.doi.org/10.1145/2568225.2568266

[15] Victor M. González and Gloria Mark. 2004. "Constant, Constant, Multi-Tasking Craziness": Managing Multiple Working Spheres. In *Proceedings of the 2004 Conference on Human Factors in Computing Systems - CHI '04 (CHI '04)*. ACM Press, Vienna, Austria, 113–120. DOI: http://dx.doi.org/10.1145/985692.985707

[16] David M. Hilbert and David F. Redmiles. 2000. Extracting Usability Information from User Interface Events. *Comput. Surveys* 32, 4 (Dec. 2000), 384–421. DOI:http://dx.doi.org/10.1145/371578.371593

[17] Dugald Ralph Hutchings, Greg Smith, Brian Meyers, Mary Czerwinski, and George Robertson. 2004. Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users. In *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '04*. ACM Press, Gallipoli, Italy, 32. DOI: http://dx.doi.org/10.1145/989863.989867

[18] Shamsi T. Iqbal, Xianjun Sam Zheng, and Brian P. Bailey. 2004. Task-Evoked Pupillary Response to Mental Workload in Human-Computer Interaction. In *Extended Abstracts of the 2004 Conference on Human Factors and Computing Systems - CHI '04*. ACM Press, Vienna, Austria, 1477. DOI: http://dx.doi.org/10.1145/985921.986094

[19] Mikkel Rønne Jakobsen and Kasper Hornbæk. 2010. Piles, Tabs and Overlaps in Navigation among Documents. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*. ACM Press, Reykjavik, Iceland, 246. DOI: http://dx.doi.org/10.1145/1868914.1868945

[20] Steven Jeuris, Paolo Tell, Steven Houben, and Jakob E. Bardram. 2018. The Hidden Cost of Window Management. *CoRR* abs/1810.04673 (2018).

[21] Mik Kersten and Gail C. Murphy. 2005. Mylar: A Degree-of-Interest Model for IDEs. In *Proceedings of the 4th International Conference on Aspect-Oriented Software Development - AOSD '05*. ACM Press, Chicago, Illinois, 159–168. DOI: http://dx.doi.org/10.1145/1052898.1052912

[22] Jeff Klingner. 2010. Fixation-Aligned Pupillary Response Averaging. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications - ETRA '10*. ACM Press, Austin, Texas, 275. DOI: http://dx.doi.org/10.1145/1743666.1743732

[23] Sophie Leroy. 2009. Why Is It so Hard to Do My Work? The Challenge of Attention Residue When Switching between Work Tasks. *Organizational Behavior and Human Decision Processes* 109, 2 (2009), 168–181. DOI:http://dx.doi.org/10.1016/j.obhdp.2009.04.002

[24] Paul Luo Li, Andrew J. Ko, and Jiamin Zhu. 2015. What Makes a Great Software Engineer?. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE '15)*. IEEE Press, IEEE, Florence, Italy, 700–710. DOI: http://dx.doi.org/10.1109/ICSE.2015.335

[25] Gloria Mark, Mary Czerwinski, and Shamsi T. Iqbal. 2018. Effects of Individual Differences in Blocking Workplace Distractions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, Montreal QC, Canada, 1–12. DOI: http://dx.doi.org/10.1145/3173574.3173666

[26] Gloria Mark, Victor M. Gonzalez, and Justin Harris. 2005. No Task Left Behind?: Examining the Nature of Fragmented Work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '05*. ACM Press, Portland, Oregon, USA, 321. DOI: http://dx.doi.org/10.1145/1054972.1055017

[27] Gloria Mark, Daniela Gudith, and Ulrich Klocke. 2008. The Cost of Interrupted Work: More Speed and Stress. In *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08*. ACM Press, Florence, Italy, 107. DOI: http://dx.doi.org/10.1145/1357054.1357072

[28] Gloria Mark, Shamsi Iqbal, and Mary Czerwinski. 2017. How Blocking Distractions Affects Workplace Focus and Productivity. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers on - UbiComp '17*. ACM, ACM Press, Maui, Hawaii, 928–934. DOI: http://dx.doi.org/10.1145/3123024.3124558

[29] Gloria Mark, Shamsi Iqbal, Mary Czerwinski, and Paul Johns. 2015. Focused, Aroused, but so Distractible: Temporal Perspectives on Multitasking and Communications. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, Vancouver, BC, Canada, 903–916. DOI: http://dx.doi.org/10.1145/2675133.2675221

[30] Stephanie McMains and Sabine Kastner. 2011. Interactions of Top-down and Bottom-up Mechanisms in Human Visual Cortex. *Journal of Neuroscience* 31, 2 (2011), 587–597. DOI: http://dx.doi.org/10.1523/JNEUROSCI.3766-10.2011

[31] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz. 2017. The Work Life of Developers: Activities, Switches and Perceived Productivity. *IEEE Transactions on Software Engineering* 43, 12 (Dec. 2017), 1178–1193. DOI: http://dx.doi.org/10.1109/TSE.2017.2656886

[32] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*. ACM Press, Hong Kong, China, 19–29. DOI: http://dx.doi.org/10.1145/2635868.2635892

[33] Andre N. Meyer, Gail C. Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. Design Recommendations for Self-Monitoring in the Workplace: Studies in Software Development. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (Dec. 2017), 1–24. DOI:http://dx.doi.org/10.1145/3134714

[34] André N. Meyer, Sebastian Müller, Manuela Züger, and Jan Pilzer. 2020. PersonalAnalytics. https://github.com/sealuzh/PersonalAnalytics. (2020). [Accessed January 8, 2020].

[35] Roberto Minelli, Andrea Mocci, and Michele Lanza. 2015a. I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time. In *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, Florence, Italy, 25–35. DOI: http://dx.doi.org/10.1109/ICPC.2015.12

[36] Roberto Minelli, Andrea Mocci, and Michele Lanza. 2015b. The Plague Doctor: A Promising Cure for the Window Plague. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension (ICPC '15)*. IEEE Press, Piscataway, NJ, USA, 182–185. DOI:http://dx.doi.org/10.1109/ICPC.2015.28

[37] G.C. Murphy, M. Kersten, and L. Findlater. 2006. How Are Java Software Developers Using the Elipse IDE? *IEEE Software* 23, 4 (July 2006), 76–83. DOI: http://dx.doi.org/10.1109/MS.2006.105

[38] Marketta Niemelä and Pertti Saariluoma. 2003. Layout Attributes and Recall. *Behaviour & information technology* 22, 5 (2003), 353–363. DOI: http://dx.doi.org/10.1080/0144929031000156924

[39] Nuria Oliver, Mary Czerwinski, Greg Smith, and Kristof Roomp. 2008. RelAltTab: Assisting Users in Switching Windows. In *Proceedings of the 13th International Conference on Intelligent User Interfaces - IUI '08*. ACM Press, Gran Canaria, Spain, 385. DOI: http://dx.doi.org/10.1145/1378773.1378836

[40] Nuria Oliver, Greg Smith, Chintan Thakkar, and Arun C. Surendran. 2006. SWISH: Semantic Analysis of Window Titles and Switching History. In *Proceedings of the 11th International Conference on Intelligent User Interfaces - IUI '06*. ACM Press, Sydney, Australia, 194. DOI:http://dx.doi.org/10.1145/1111449.1111492

[41] George Robertson, Eric Horvitz, Mary Czerwinski, Patrick Baudisch, Dugald Hutchings, Brian Meyers, Daniel Robbins, and Greg Smith. 2004. Scalable Fabric: Flexible Task Management. In *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '04*. ACM, ACM Press, Gallipoli, Italy, 85–89. DOI: http://dx.doi.org/10.1145/989863.989874

[42] David Roethlisberger, Oscar Nierstrasz, and Stephane Ducasse. 2009. Autumn Leaves: Curing the Window Plague in IDEs. In *2009 16th Working Conference on Reverse Engineering*. IEEE, Lille, France, 237–246. DOI:http://dx.doi.org/10.1109/WCRE.2009.18

[43] Robert D Rogers and Stephen Monsell. 1995. Costs of a Predictible Switch between Simple Cognitive Tasks. *Journal of experimental psychology: General* 124, 2 (1995), 207. DOI: http://dx.doi.org/10.1037/0096-3445.124.2.207

[44] Joshua S Rubinstein, David E Meyer, and Jeffrey E Evans. 2001. Executive Control of Cognitive Processes in Task Switching. *Journal of experimental psychology: human perception and performance* 27, 4 (2001), 763. DOI:http://dx.doi.org/10.1037/0096-1523.27.4.763

[45] Heider Sanchez, Romain Robbes, and Victor M. Gonzalez. 2015. An Empirical Study of Work Fragmentation in Software Evolution Tasks. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Montreal, QC, Canada, 251–260. DOI: http://dx.doi.org/10.1109/SANER.2015.7081835

[46] Bonita Sharif and Jonathan I. Maletic. 2010a. An Eye Tracking Study on camelCase and Under_score Identifier Styles. In *2010 IEEE 18th International Conference on Program Comprehension*. IEEE, Braga, Portugal, 196–205. DOI: http://dx.doi.org/10.1109/ICPC.2010.41

[47] Bonita Sharif and Jonathan I. Maletic. 2010b. An Eye Tracking Study on the Effects of Layout in Understanding the Role of Design Patterns. In *2010 IEEE International Conference on Software Maintenance*. IEEE, Timi oara, Romania, 1–10. DOI: http://dx.doi.org/10.1109/ICSM.2010.5609582

[48] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. 1997. An Examination of Software Engineering Work Practices. In *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON '97)*. IBM Press, Toronto, Ontario, Canada, 21–.

[49] Greg Smith, Patrick Baudisch, George Robertson, Mary Czerwinski, Brian Meyers, and Daniel Robbins. 2003. GroupBar: The TaskBar Evolved. In *(2003) OZCHI 2003 Conference for the Computer-Human Interaction Special Interest Group of the Human Factors Society of Australia*. University of Queensland, Brisbane, Australia.

[50] Susanne Tak and Andy Cockburn. 2010. Improved Window Switching Interfaces. In *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '10*. ACM Press, Atlanta, Georgia, USA, 2915. DOI: http://dx.doi.org/10.1145/1753846.1753884

[51] Susanne Tak, Andy Cockburn, Keith Humm, David Ahlström, Carl Gutwin, and Joey Scarr. 2009. Improving Window Switching Interfaces. In *Human-Computer Interaction – INTERACT 2009*, Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler (Eds.). Vol. 5727. Springer Berlin Heidelberg, Berlin, Heidelberg, 187–200. DOI: http://dx.doi.org/10.1007/978-3-642-03658-3_25

[52] Tobii. 2020. Tobii Eye Tracker 4C. https://gaming.tobii.com/tobii-eye-tracker-4c. (2020). [Accessed January 8, 2020].

[53] Vincent W.-S. Tseng, Matthew L. Lee, Laurent Denoue, and Daniel Avrahami. 2019. Overcoming Distractions during Transitions from Break to Work Using a Conversational Website-Blocking System. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–13. DOI: http://dx.doi.org/10.1145/3290605.3300697

[54] Manuela Waldner, Markus Steinberger, Raphael Grasset, and Dieter Schmalstieg. 2011. Importance-Driven Compositing Window Management. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems - CHI '11*. ACM Press, Vancouver, BC, Canada, 959. DOI: http://dx.doi.org/10.1145/1978942.1979085

[55] Andrew Warr, Ed H. Chi, Helen Harris, Alexander Kuscher, Jenn Chen, Robert Flack, and Nicholas Jitkoff. 2016. Window Shopping: A Study of Desktop Window Switching. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, Santa Clara, California, USA, 3335–3338. DOI:http://dx.doi.org/10.1145/2858036.2858526

[56] S. Yusuf, H. Kagdi, and J.I. Maletic. 2007. Assessing the Comprehension of UML Class Diagrams via Eye Tracking. In *15th IEEE International Conference on Program Comprehension (ICPC '07)*. IEEE, Banff, Alberta, BC, 113–122. DOI: http://dx.doi.org/10.1109/ICPC.2007.10